UCRL-JC-122246

Using Commercial-Off-the-Shelf (COTS) Software in High-Consequence Safety Systems

Prepared by
J. A. Scott
G. G. Preckshot
J. M. Gallagher

FESSP
Fission Energy and Systems Safety Program

Lawrence Livermore National Laboratory

Using Commercial-Off-the-Shelf (COTS) Software in High-Consequence Safety Systems

Manuscript date: November 10, 1995

Prepared by
J. A. Scott
G. G. Preckshot

Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550

J. M. Gallagher

U.S. Nuclear Regulatory Commission

Using Commercial Off-the-Shelf (COTS) Software in High-Consequence Safety Systems

John A. Scott and G. Gary Preckshot, Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550

John M. Gallagher, U.S. Nuclear Regulatory Commission
Mail Stop 8 H3, NRR/HICB
Washington, DC 20555-0001

ABSTRACT

This paper is based on work performed by Lawrence Livermore National Laboratory[1] to assist the U.S. Nuclear Regulatory Commission in understanding the state of the art with respect to applying commercial off-the-shelf (COTS) software to high-consequence safety systems. These systems, for which the consequences of failure can be severe or catastrophic, must be developed, implemented, and maintained in ways that provide assurance that catastrophic consequences will be prevented. This paper discusses various aspects of the question of using commercially available software in these systems. Risk, grading, and system assessment are discussed, and relevant standards are summarized. A recommendation for addressing key issues regarding the use of commercial software in high-consequence safety systems is given.

I. INTRODUCTION

Systems for which the consequences of failure can be severe or catastrophic, i.e., high-consequence safety systems (HCSS's), must be developed, implemented, and maintained in ways that provide assurance that catastrophic consequences will be prevented. The process of assurance may include avoiding faults in the system, designing the system to tolerate faults without progressing to severe consequences, or providing additional measures to mitigate the consequences of system failure. Developers of such systems focus on achieving safety goals even if extensive resources must be expended. Historically, guidance for developing software to be used in an HCSS has been based on custom builds using state-of-the-art software and safety engineering practices, both at the system and software level. Such development processes can be quite costly.

Recently, economic pressures have caused a trend toward streamlining organizations and increasing their efficiency. System development activities, particularly those associated with HCSS's, are expensive and, therefore, are the focus of efforts to increase efficiency and reduce costs. This is not merely an attempt to increase or protect profit—highly trained personnel are a scarce resource. More efficient allocation of skilled professionals to the unsolved safety problems rather than those already solved results in safer systems.

Reflecting the scarcity of skilled talent, the costs associated with mass-produced electronic hardware have been decreasing steadily for many years, but costs for labor-intensive software development have been continually increasing. Consequently, software costs account for a growing percentage of system development costs, which creates considerable pressure to improve the efficiency and productivity of software developers.

Software reuse is one means for increasing software development productivity. The effective use of subroutine libraries has long been a goal of software development organizations. More recently, the advent of object-oriented techniques has facilitated the development of reusable code. Efforts to achieve reusability on a larger scale are also underway. One example is the Asset Source for Software Engineering Technology (ASSET) facility established under ARPA's Software Technology for Adaptable, Reliable Systems (STARS) program. One of ASSET's goals is to provide a national marketplace for reusable software products.[2]

Using commercially available software is another form of software reuse and is the subject of this paper. In many low-risk applications, it is practical to incorporate this type of software into systems. In a number of other potential applications of commercially available software, e.g., reactor safety systems, it requires a considerable leap of faith to believe that safety systems developed by knowledgeable reactor instrumentation engineers can be replaced by commercial off-the-shelf (COTS) software. Yet the modern trend toward more and more engineering being done in software packages makes this an inevitable suggestion.

A. *Why There is a Problem*

If this were the best of all possible worlds, commercial software would be developed by professionals of calm and deliberate mien in organizations with realistic schedules and budgets. However, the term "COTS software" is generally antithetical to realistic schedules and budgets, and COTS products are often developed in an atmosphere of haste, insufficient budget, and ever-increasing "features" to attract and hold a larger market share. Such products are reasonably

[1] This work was supported by the United States Nuclear Regulatory Commission under a Memorandum of Understanding with the United States Department of Energy, and performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

[2] Personal communication from Katherine Bean at the ASSET facility, Jan. 10, 1995.

priced, but access to development and product verification documentation, if any was produced, is usually limited. Some products are developed specifically for the HCSS market, although the selection is limited. The problem is to distinguish these, and other competently developed products, from those that are just “pretty good.” The payoff, for a good choice, is releasing professionals from the task of developing a product that recapitulates the COTS product, and applying them to the real problem: safe implementation of a high-consequence system.

B. Risk

A central theme in this exposition is risk. Treatment of risk is one of the differences between high-consequence safety systems and those of lower consequence. Risk, or risk exposure, is usually taken to be the product of consequence times the probability of occurrence. For low-consequence systems, risk exposure is linear in both variables. However, for high-consequence systems, risk may be non-linear, or undefined but huge in the public eye. There are two reasons for this. The first is that the extent of a high-consequence failure is difficult to predict. The second is that estimates of low probabilities for “rare” events are usually optimistic. Human beings are constantly surprised to discover interdependencies between things they thought were independent. The implication for COTS software used in HCSS's is that considerably higher standards of proof of suitability will be required than would normally be the case for normal industrial hazards. This makes risk grading a useful gauge for COTS assessment effort.

Another implication that risk exposure has for HCSS COTS software usage, is that any architecture using COTS products must be tolerant of product failure. This is true for any software product used in this venue, because nobody has yet discovered how to build error-free software, or, for that matter, write error-free specifications. However, COTS products face additional hurdles not encountered by software developed specifically for an HCSS. The specifically developed software is tailored to the application, falls under the same engineering controls as the target HCSS system, and is designed for maintainability by the target industry. In contrast, COTS products are usually not designed for narrow applications, but instead must appeal to a broader market by commercial necessity. There is usually a target price, a time-constrained market window, and a set of features to be delivered for the target price. To broaden the appeal of the product, it is often multi-configurable, and the developer is responsive to the demands of the most lucrative segment of his clientele. Compared to the specifically developed product, the COTS product is more likely to have extra features that may surface as unused or unintended functions in the HCSS application. In high-risk HCSS applications, therefore, COTS products cannot be the last line of defense.

C. System-Level COTS Product Assessment

The issues facing a designer of an HCSS who would like to use a COTS product fall into a natural order if risk identification is used to grade the rigor of the assessment process. One of the goals of any assessment process should be to eliminate unsuitable options as soon as possible, reducing time spent on untenable candidates. This is best done starting from the viewpoint of the overall system in which the COTS product will be embedded, then deriving the functional requirements the product must satisfy, and finally proceeding to product-specific issues. Risk identification comes first, followed by product identification, and determination of level of rigor of assessment steps that follow.

The first step taken with any HCSS design should be some sort of hazard analysis that identifies potential high-consequence failures. For example, an architecture is proposed in which one or more COTS products play a role in detecting, preventing, or mitigating failures. Probabilistic risk assessment (PRA) is a typical type of analysis that is used to identify the functions the system is required to perform. This analysis uses fault trees to model subsystems, and event trees to model accident progressions. One of the outcomes of the analysis should be identification of the safety functions for which specific subsystems are responsible. Some of these subsystems may incorporate COTS products, and the safety functions the products must provide become the basis for assessment of candidate products. The PRA should also determine whether diverse means will be necessary to ensure the performance of a critical set of safety functions.

Once the environment for the COTS product has been established in terms of the safety functions to be performed and the nature of the product’s interfaces to other system components, the next step is to assess the COTS product itself in order to establish sufficient confidence that it will perform the intended safety functions, and that it will interface with other system components in ways that will not lead to unsafe conditions. An example process is given in the Appendix. It is crucially important at this point that the COTS product be under strict configuration control. The product must be precisely identified and all pertinent documentation must be clearly linked to the exact version and configuration of the product being assessed. The assessment cannot be done on a product that is essentially a “moving target.” With respect to the performance of safety functions, the assessment seeks to determine that the product characteristics provide sufficient confidence that the functions will be accomplished. Quality attributes such as correctness, performance, and reliability are assessed. With respect to system interactions, it is necessary to determine that the product will not interfere with the performance of system safety functions. Interference could occur because of abnormal behavior within the COTS product or because of inappropriate responses to external abnormal conditions or events. The presence of unused functions (legitimate features of the COTS product that are not needed for the safety-related application) in most COTS products requires that special assessments be performed to ensure that inadvertent activation of these functions is prevented or, if they are activated, that they do not interfere with system safety functions.

D. *The Regulator's Dilemma*

The general goal of regulatory bodies is to ensure that adequate levels of safety are achieved when high-consequence systems within their purview are developed. For software, considerable effort has been expended to establish guidelines and requirements for the application of maturing software engineering practices to new developments. Since, in most cases, software development methods cannot guarantee the absence of faults, these guidelines and requirements call for comprehensive, extensive application of currently accepted software engineering practices to HCSS's. In cases with potential catastrophic consequences, additional systems measures such as diversity and defense-in-depth are employed.

The pressures to employ COTS software in these systems creates a new set of difficult questions for regulators. Aside from technical questions related to the safe integration of a COTS software product into an HCSS, frequent difficulties are also encountered due to various competitive or management concerns. These concerns tend to interfere with access to records and information needed for product assessment. If the desired records and information are unavailable, the regulator (and developer) is then faced with either rejecting the COTS product for use in the HCSS or with making a determination about the acceptability of a COTS software product based on acceptance criteria that are different than those used for new developments. There is a danger in the search for alternatives. It must be shown convincingly that alternative acceptance criteria provide confidence equivalent to that obtained from the processes applied to new developments since, if it is possible to achieve this confidence with less costly new alternatives, the existing processes used in new developments will quickly be abandoned. Unhappily, the search for alternatives to the assurance techniques used for new developments frequently leads to issues that are still subjects of research in software engineering.

II. How Much Assessment Effort?

A. *Grading*

Efficiency dictates that efforts to achieve safety should be commensurate with risk exposure. The rationale for grading is that extraordinary efforts should not be made to manage an insignificant risk, but if a significant risk is present, appropriate efforts should be expended to manage that risk. Various types of risk can be considered with respect to grading. Two types of grading of interest for HCSS's are consequence grading, which is based on the impact associated with potential failure, and grading with respect to the relative importance to safety of constituent components. The former is related to the overall extent of risk reduction effort necessary to prevent the realization of hazards. Within the context of the consequence category, the latter is related to the maximum allowable probability of component or subsystem failure. Four consequence levels are typically recognized:

- Failures that show the potential for significant off-site consequences, such as consequences having a major impact on the public.
- Failures that show the potential for significant on-site consequences. Possible examples are consequences involving on-site fatalities, loss of mission, or economic disaster for the owner/operator.
- Failures that show the potential for only localized consequences. Possible examples are consequences involving a small number of recoverable injuries or survivable economic impact on the owner/operator.
- Those having negligible impact, such as minor delays or inconveniences.

For HCSS's, the highest category is usually the one of interest. When grading components or subsystems with respect to relative importance to safety, consideration of the specific application area might allow specific types of subsystems to be placed in categories for which varying levels of assurance would be required. An example of grading with respect to importance to safety is found in the IEC 1226[3] standard in which categories A, B, C, and Unclassified are established for types of systems found in nuclear power plants. Examples of systems in each category are, respectively,

- Safety shutdown systems
- Control systems
- Some warning and alarm systems
- Non-safety systems.

B. *Benefits of Grading*

The general benefit of applying a valid and effective grading process is that each system will receive the level of assurance effort that is appropriate for that system. Systems that have some importance to safety will be treated as such and not as uncontrolled developments. Systems that are unimportant to safety will not have excessive assurance efforts applied.

Grading is especially important with respect to determining the acceptability of COTS products. As the magnitude of potential consequences decreases or as the safety role of a particular subsystem decreases, the potential impact of unknowns related to the alternative acceptance criteria discussed above is lessened. Therefore, the developer and regulator have more flexibility with respect to the basis for acceptance of the COTS software product.

[3] IEC 1226, "Nuclear Power Plants—Instrumentation & Control Systems Important for Safety-Classification," International Electrotechnical Commission, 1993.

III. The Assessment Process

A. *The Role of Standards*

Software standards tend to represent consensus thinking with respect to software issues. This consensus includes the viewpoints of researchers, who view the issues from a theoretical standpoint, and practitioners, who view the issues in terms of day-to-day realities. Regulators have specific goals with respect to safety and need to be aware of both viewpoints. Since the production of fault-free software is still not possible in most cases, the standards provide important information regarding accepted software practices that form a basis for acceptance criteria for software, both for new developments and for COTS software (although the former has been treated far more extensively). These standards also help to identify software issues that are currently research questions. Descriptions of some of the key standards, with an emphasis on the nuclear industry, are given below.

IEEE 7-4.3.2-1993—Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations

This standard has a section dealing with COTS software and addresses testing for COTS items. It also considers software development methods and operating experience. The standard has a subjective nature, however, as evidenced by the following:

> "Exceptions to the development steps required by this standard or referenced documents may be taken as long as there are other compensating factors that serve to provide equivalent results."
>
> "Acceptance shall be based upon an engineering judgment that the available evidence provides adequate confidence that the existing commercial computer, including hardware, software, firmware, and interfaces, can perform its intended functions."

While the general intent of these passages is clear, there is room for a varying strictness of interpretation. For professional engineers considering the use of COTS software in an HCSS, it would probably be assumed that it must be explicitly and convincingly shown how information from a compensating factor provides equivalent results and, when engineering judgment is used, that it be applied to specific, narrowly defined questions and that its basis be convincing and documented.

IEC 880—Software for Computers in the Safety Systems of Nuclear Power Stations

IEC 880 is a prescriptive standard which offers detailed criteria that software under its purview must satisfy. Risk-related requirements are emphasized, as are interfaces with and relations to other systems and hardware. The following five-point summation of Section 5 of IEC 880 illustrates the risk-based approach:

- Safety relevance of software parts should be determined
- More limiting recommendations apply to risky parts
- High-safety-impact software modules should be easily identifiable from system structure and data layout
- Available testing and validation procedures should be considered when doing the design
- If difficulties arise, a retrospective change of style may be required.

"Self supervision" is required, meaning that the software includes code to detect hardware errors and errors committed by software. Self supervision is only regarded in the literature as effective for detecting hardware errors; considerable controversy still exists on whether there are effective means of detecting software errors with more software.

Draft Addition (to IEC 880) on Commercial Dedication

IEC 880 provides a strong connection between either risks or safety considerations and software (or system) requirements, and this connection is continued and enhanced in the draft addition on commercial dedication. The draft addition strongly emphasizes determining the safety functions that a COTS product will perform before deciding on the rigor of the commercial dedication process to be followed. This is combined with a strict view of experience data; for important safety functions, COTS experience data must be relevant and statistically valid.

IEC 1226—The Classification of Instrumentation and Control Systems Important to Safety for Nuclear Power Plants

IEC 1226 provides the missing link that the other standards discussed herein lack (at least for the nuclear industry): a consistent definition of safety categories. This standard uses terms familiar to those involved in nuclear power plant safety: redundancy, diversity, defense-in-depth, and reliability.

IEEE Software Standards

The IEEE software standards are widely recognized consensus standards addressing a range of software issues. Some of the key standards related to the assessment of COTS software are those addressing configuration management, verification and validation (V&V), software development and safety plans, and life-cycle processes. The guidance in these standards generally addresses new developments, but this information is valuable in assessing software processes and the product development records of the COTS software product.

ISO 9000-3

The ISO 9000 standards apply to quality assurance programs in general, and are not limited to software. ISO 9000-3 interprets the general standards as applied to software, and fulfills somewhat the same role as IEEE 730.1 (Software Quality Assurance Plans); that is, it is a pro forma standard that acts, in part, as an umbrella standard, mentioning other aspects of software quality such as software configuration management and software V&V.

B. Difficult Issues in Assessing COTS Software

A number of issues that are particularly difficult to address have surfaced with respect to the determination of the acceptability of a COTS software product for use in an HCSS. Brief discussions of the issues and key aspects are given below.

Unexpected Functions

Unexpected functions are either unused functions (functions intended by the developer but not needed for the application) or unintended functions (unplanned functions arising from design or implementation errors). Unexpected functions are functions built into the COTS software product either as intended commercial features or as design errors, neither of which are desired for the high-consequence application. The probability that unexpected functions would be present in a new software development for a specific application is much less than for the COTS software product. The concern is that unexpected functions might affect the performance of system safety functions if the functions are inadvertently activated. unexpected functions are a concern in system-level COTS product assessment, as discussed above.

Operational Profiles

In order to evaluate the COTS software in performing its assigned safety functions and interacting with the overall system, it is necessary to understand the operational profile of the software's role in the system. This profile includes items such as the nature of inputs and outputs, transaction loads, performance requirements, and reliability requirements. For new developments, this information is included in the product specifications and the product is designed to fit the profile. For COTS products, the operational profile assumed in its design might not be known precisely. Consequently, it is difficult to assess where discrepancies between the design and the specification for COTS usage in an HCSS might appear. This could be particularly important in situations where the software must operate outside of its normal parameters.

Version Control

Two key considerations arise regarding version control. The first is a COTS product to be evaluated for use in an HCSS must be identified by version. All of the information pertaining to the COTS item to be used in the assessment must be clearly linked to the specific version to be incorporated into the HCSS. Different versions require separate assessments. It is also possible that a COTS product is so highly configurable that information from several sources regarding the same version will not be equally valid for the COTS acceptability determination.

The second consideration is the question of how bug fixes and new releases will be handled once the COTS software has been integrated into a system. Also, who will perform the modifications, and how will the acceptability of modified software be determined? These are particularly troublesome questions in cases where software was accepted under alternative acceptance criteria because product information was not available. It may be possible to freeze versions for a period of time, but eventually, system evolution or vendor support requirements will generate a need for changes.

Site-Specific Configuration

The COTS product must be evaluated in the light of a full range of system configuration questions. New developments would be designed for the specific HCSS site configuration requirements. Considerations include site requirements and constraints, platform configurations, configurations of layered or interfacing software, and possible configurations of the COTS product itself.

Error Reporting

Error reporting is a key issue with respect to the use of commercial software in safety-related systems. Two-way communication between the COTS software developer and COTS software users is necessary to ensure that known or recently discovered errors are documented for evaluation of their potential impact on high-consequence systems, and to ensure that additional information can be obtained from the software developer if anomalies are observed during system operations. Long-term relationships of this type can be difficult to establish and maintain with vendors of commercial products.

C. Process or Product?

The Case for Product

Careful examination of a COTS product is essential to assess its acceptability for use in an HCSS. Three sources for information should be used in the assessment: product development records, operating experience data, and post-development product testing. Product development records include specifications, code, V&V records, etc. Experience data can provide information about reliability, correctness, and performance; however, care must be taken to ensure that the data are valid. Data based on other versions or different configurations may not be applicable. Post-development product testing can be performed to obtain additional assessment data. This testing might be done to address minor deficiencies in development records or to address special requirements associated with the planned use of the COTS software in an HCSS.

The Case for Process

It is also important to assess the development processes that were applied during product development. This assessment helps to establish the validity and value of the product development records. Product information is not very useful without confidence that the processes that generated the information are comparable to commonly accepted practices for developing high-integrity software. Various methods have been developed to assess these processes, including ISO and SEI CMM (Software Engineering Institute Capability Maturity Model) style assessments, and other assessment processes that may have special focuses. In addition to software development process assessment, a number of organizational indicators can also provide important information. It is important to note that certifications from past assessments are perishable and that those past assessments might not encompass everything of interest with respect to COTS acceptability determinations.

IV. Replacing Missing Documentation

Considerable controversy rages about the acceptability of various approaches to replacing, reconstructing, or substituting for missing COTS product documentation. Some viewpoints, such as that found in IEEE 7-4.3.2, focus on the question of applying engineering judgment to these and other assessment questions. IEEE 7-4.3.2 was written for competent engineers, and in that light, it is fair to assume that engineering judgment will be applied carefully to specific, narrowly defined questions, and that the rationale for the judgment will be documented and able to withstand critical, external scrutiny. Engineering judgment cannot be used to justify generic, hand-waving arguments about COTS product acceptability.

One proposed alternative to the lack of documentation is operational experience. It seems intuitive that experience data derived from extensive usage of the same version of a product in similar applications would indicate that the product is acceptable for the intended application. There are several problems with this assumption. First, configuration data is rarely supplied and version data is often missing. Therefore, the statistical validity of the data is unknown. Large bases of experience data usually span releases and configurations of a given product. The second problem is that circumstances surrounding the occurrence, monitoring, and recording of failures are often vaguely reported. These activities are not uniformly controlled; in fact, there might be some motivation for COTS vendors to limit publication of negative experience. Another key issue regarding operating experience is that extensively used products can still have crucial faults that could cause problems in safety systems. The tendency is to consider operational experience to be like extensive random testing. In this regard, operational experience suffers from the same shortcoming as testing: testing cannot prove the absence of faults.

A second proposed alternative acceptance criterion is using post-development testing to compensate for a major lack of product development documentation. However, as noted above, testing cannot confirm the absence of faults in software. In addition, it is difficult to use testing to detect unintended functions. Testing can, however, demonstrate that intended functions are implemented and that anticipated error conditions are handled properly.

A third approach to replacing missing data is to perform data reconstruction with methods such as reverse engineering. This can be a difficult task requiring as much effort as doing a new development and, even if accomplished, it is not clear that the reconstruction process was error-free.

In general, the approach of compensating for missing information by using information from other sources involves research questions or data validity problems. One must know exactly how one source of information replaces information from another. Until this is known, there is considerable risk in accepting a COTS product for use in an HCSS based on such techniques.

However, compensation is more acceptable as risk exposure decreases.

V. The Bottom Line

Given the current state of the art in software engineering and the technical, political, and maintenance considerations associated with typical commercial software, it appears that it will be difficult at best to incorporate COTS software products into high-consequence safety systems. If an effective grading process is in place, however, the tradeoffs associated with the use of COTS products in lower risk categories become more palatable.

For software in the highest risk categories, our recommendation is that, after the systems analyses have been performed and the COTS usage specifications developed, the acceptance process be based primarily on an examination of comprehensive product development records. Assessments of the COTS vendor’s software processes should also be performed to ensure that the product development records accurately represent the quality of the software. Post-development testing should be performed to address minor deficiencies in product development records and to address unique requirements that might arise from a particular proposed use of the product. Operational experience, if statistically valid, can be used to increase confidence in the product and to help answer questions arising in examining specific aspects of the software. In this software category then, acceptance should be based on a combination of information from all sources, but the use of compensating factors should be minimized, if used at all. This approach is consistent with current requirements for new developments of this type of software and, therefore, will not create a situation in which the COTS acceptance process serves as a conduit for escaping the scrutiny of new development requirements.

A. Is COTS Really Cost-Effective?

Acceptance

Although it appears initially to be much cheaper to purchase COTS software than to perform new developments, the costs associated with an acceptance process that attempts to provide adequate assurance for software used in systems in the highest risk categories can be quite high. Less expensive "shortcuts" provided by proposed alternative acceptance criteria have not been demonstrated to provide adequate assurance. In typical cases, it can be difficult and expensive to obtain comprehensive product development records, not to mention vendor process assessments. Ensuring that unintended functions cannot impair system safety can also be expensive.

Finally, commercial vendors are usually not motivated to enter into long-term error reporting contracts or to assume liability for such reporting, so establishing such arrangements might also be quite expensive.

Maintenance

Accepting a COTS software product for use in an HCSS is just the beginning. Users will eventually be forced into upgrades, and an upgraded commercial product must be re-qualified. This process is probably easier if the upgrades are produced by a known, high-quality development process, preferably the same one that was used for the original development. If the COTS software had been approved by an alternative process because development records were not available, the question of upgrades might be particularly vexing, especially if the original vendor is no longer in business. User modification, even if full documentation is available, will probably be more difficult than if done by the original developing organization. The costs associated with these difficulties can be very high and should be considered when making the initial decision to use COTS software in high-consequence safety systems.

Acceptance and maintenance issues, and their associated costs, become less important as importance to safety decreases.

B. Is it Possible to Use COTS Software in an HCSS?

Yes. Some software vendors, such as programmable logic controller vendors, produce software with the knowledge that it will be used in systems with medium to high risks. Some of these vendors use software processes that have been designed to produce high-integrity software. They are generally aware of the types of hazards associated with the systems in which their products will be used and those hazards have been considered in their designs. In order to meet the demands of the high-integrity marketplace, they may be motivated to form long-term partnerships with users and to supply additional reliability documentation. Such vendors may well be in a position to meet applicable acceptance and regulatory requirements for the use of their products in HCSS's.

VI. References

[1] J. Dennis Lawrence, *Software Reliability and Safety in Nuclear Reactor Protection Systems,* NUREG/CR-6101, UCRL-ID-114839, Lawrence Livermore National Laboratory, 1993.

[2] J. Dennis Lawrence and W. L. Persons, *Survey of Industry Methods for Producing Highly Reliable Software*, NUREG/CR-6278, UCRL-ID-117524, Lawrence Livermore National Laboratory, 1994.

[3] J. Dennis Lawrence and G. G. Preckshot, *Design Factors for Safety-Critical Software*, NUREG/CR-6294, Lawrence Livermore National Laboratory, 1994.

[4] G. G. Preckshot, *Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems*, NUREG/CR-6303, Lawrence Livermore National Laboratory, 1994.

APPENDIX: QUALIFICATION CRITERIA

A representative risk-based COTS software acceptance process is proposed in this appendix. Tables A-1 through A-3 describe a safety category grading scheme. Table A-1 describes safety categories of systems used in nuclear reactors. Table A-2 describes categories of usage, based on whether the COTS software is itself used, or products of the COTS software are used. Table A-3 gives the resultant safety categories.

Table A-4 describes the initial four steps of the risk-based acceptance process. These steps determine which of the following three tables are entered. Tables A-5 through A-7 describe the level of rigor with which to apply Tables A-8 through A-21.

Tables A-8 through A-21 give standards-derived acceptance criteria.

Table A-1
Safety Categories

Category	**Example Systems**
A	Reactor Protection System (RPS) Engineered Safety Features Actuation System (ESFAS) Instrumentation essential for operator action
B	Reactor automatic control system Control room data processing system Fire suppression system Refueling system interlocks and circuits
C	Alarms, annunciators Radwaste and area monitoring Access control system Emergency communications system

Table A-2
COTS Usage Categories

Usage Category	**Description**	**Equivalent IEC 1226**
Direct	Directly used in an A, B, or C application.	Usage
Indirect	Directly produces executable modules that are used in A, B, or C applications (software tools such as compilers, linkers, automatic configuration managers, or the like). Produces A modules Produces B modules Produces C modules	 A or B [4] B or C [5] unclassified
Support	CASE systems, or other support systems that indirectly assist in the production of A, B, or C applications, or software that runs as an independent background surveillance system of A, B, or C applications.	unclassified
Unrelated	Software that has no impact on A, B, or C applications.	unclassified

[4] The choice of A or B category depends upon whether the A module has diverse alternatives or whether there is another software tool, treated

Table A-3
COTS Safety Category Criteria

1	If the COTS product is used directly in a safety-related system, the COTS safety category is determined by the criteria of IEC 1226.
2	If the COTS product directly produces or controls the configuration of an executable software product that is used in a safety related-system and no method exists to validate the output of the COTS product, the COTS safety category is the same as that of its output, except that category C software may be produced by COTS products of the unclassified category. COTS software that directly produces category A or B software that is validated by other means is category B or C, respectively.
3	If the COTS product supports production of category A, B, or C software, but does not directly produce or control the configuration of such software modules, it is safety category unclassified.
4	If the COTS product has no impact on category A, B, or C software or systems, it is safety category unclassified.

Table A-4
Preliminary COTS Acceptance Criteria

1	Risk and hazards analyses should be used to identify system-level safety functions required.
2	The safety functions (if any) that the COTS product will perform should be identified.
3	The COTS product should be under configuration and change control. See Table A-11 for detailed software configuration management (SCM) criteria.
4	The safety category of the COTS product should be determined. Proceed to Table A-5, A-6, A-7, or A-7 depending upon category A, B, C, or unclassified, respectively.

as category A, that verifies the output of the subject tool.

[5] The choice of B or C category depends upon whether the B module has diverse alternatives or whether there is another software tool, treated as category B, that verifies the output of the subject tool.

Table A-5
Category A COTS Acceptance Criteria

5	The COTS product should have been developed under a rigorous Software Quality Assurance (SQA) Plan as defined by IEEE 730.1, ISO 9000-3, or IEC 880. This should include full V&V. See Table A-10 for detailed SQA criteria. See Table A-12 for detailed V&V criteria. See Table A-19 for minimum required V&V tasks.
6	The documentation of Table A-14 should be available for review that demonstrates Criterion 5 and also that good software engineering practices were used. Evidence should be available that the minimum required reviews of Table A-15 were conducted.
7	It should be demonstrated that the COTS product meets the requirements identified in Criterion 2.
8	It should be demonstrated that the COTS product does not violate system safety requirements or constraints.
9	The interfaces between the COTS product and other systems or software should be identified and clearly defined.
10	The COTS product should have significant (> 1 year operating time), current severe-error-free operating experience in at least two independent operating locations. Adverse reports should not be excluded even if two operating locations can be found with no adverse reports. The version and release of the proposed COTS product should be identical to that used in the experience data base. The configuration of the product in the experience data base should closely match that of the proposed COTS product.
11	All errors, severe or otherwise, should be reported and analyzed.
12	Additional validation and testing should be performed if needed to compensate for a small amount of missing documentation or alterations in configuration.

Table A-6
Category B COTS Acceptance Criteria

5	The COTS product should have been developed under a quality assurance plan and a systematic software development process. See Table A-10, entries 5 through 10 for SQA criteria. See Table A-12, entries 3 through 7 for V&V criteria.
6	Documentation should demonstrate Criterion 5. See Table A-14 for minimum required documentation.
7	It should be demonstrated that the COTS product will fulfill its safety functions as identified in Criterion 2, and that its reliability is sufficiently high that it does not present a high frequency of challenges to category A systems.
8	The COTS product is consistent with system safety requirements.
9	The COTS product has operated satisfactorily in similar applications. The version and release of reported experience may not be identical to the proposed COTS product, but a consistent configuration management program and well-managed update program provide traceability and change control.
10	Error reporting, tracking, and resolution are consistent and correctly attributable to version and release. The version and release proposed has no major unresolved problems. A current bug list should be available to COTS purchasers as a support option.

Table A-7
Category C COTS Acceptance Criteria

5	The COTS product should have been developed according to good software engineering practices. Minimum documentation as in Table A-20 should be available or reconstructable. Minimum V&V tasks as in Table A-19, entries 2, 4, 8, 9, and 19–22, should have been performed.
6	Minimum documentation described in Criterion 5, including V&V task documentation, should be available for inspection.
7	The COTS product may enhance safety by improving surveillance, improving operators' grasp of plant conditions, assisting in maintenance activities, reducing demands on category A or B systems, monitoring or reducing the effects of radiation releases, or similar purposes. The purpose or effect should be verified.
8	It should be demonstrated that the COTS product cannot adversely affect the safety functions of category A or B systems or software.
9	The COTS product has been shown to operate without serious malfunction in the instant application.
10	An error reporting scheme is planned or in place that tracks malfunctions of this COTS product in applications controlled by this applicant. Documentation and records retention allow error histories of 5 years or length of service, whichever is shorter.

Table A-8
Failure Consequence Criteria

1	Are consequences of failure unacceptable?	See Table A-9
2	Are consequences of failure acceptable?	Terminate

Table A-9
Plan Existence Criteria

1	An SQA plan and documentation exist	See Table A-10
2	A configuration management plan exists	See Table A-11
3	A software V&V plan exists	See Table A-12
4	Some of the above do not exist	See Table A-13

Table A-10
SQA Criteria

1	Does the SQA plan cover the minimum required subjects in the required format?	Format and subject matter is standard-dependent, but most standards specify similar approaches See IEEE 730.1
2	Does the plan describe responsibilities, authority, and relations between SQA units and software development units?	IEEE 730.1
3	Is minimum documentation available?	See Table A-14 for required documentation. See Table A-17 for optional documentation.
4	Were the minimum SQA reviews and audits performed?	See Table A-15 for minimum required reviews and audits
5	Are standards, practices, conventions, and metrics that were used, described?	See Table A-18 for suggested areas of standardization
6	Were procedures for problem reporting, tracking, and resolving described? Problems documented & not forgotten Problem reports validated Feedback to developer & user Data collected for metrics & SQA	IEEE 730.1 IEEE 730.2 IEEE 730.2 IEEE 730.2 IEEE 730.2
7	Were configuration management practices followed?	See Table A-11
8	Were V&V tasks performed?	See Table A-12
9	Did other software suppliers contribute to the product?	See Table A-16. “The supplier is responsible for the validation of subcontracted work.” ISO 9000-3
10	What records were generated, maintained, and retained?	IEEE 730.1
11	What methods or procedures were used to identify, assess, monitor, and control risk during development of the COTS product?	IEEE 730.1

Table A-11
Software Configuration Management Criteria

1	Does the configuration management plan cover the minimum required subjects in the required format?	Format and subject matter is standard-dependent, but most standards specify similar approaches. See IEEE 828
2	Does the plan describe responsibilities, authority, and relations between configuration management units and software development units?	IEEE 828
3	At least one configuration control board (CCB) is required. Does the plan describe the duties and responsibilities of the CCB and relations between the CCB, SQA, and software developers? e.g., Authority & responsibility Role Personnel How appointed Relation of developers & users	IEEE 828
4	Does the configuration management operation provide the following required functions? Configuration ID (baselines) Configuration control Configuration status accounting & reporting Configuration audits & reviews	IEEE 828
5	Configuration management is founded upon the establishment of "configuration baselines" for each version of each product. Is each product or version uniquely identified and "baselined"?	IEEE 828
6	Is the level of authority required for change (i.e., change control) described? Appropriate subjects include: Change approval routing lists Library control Access control R/w protection Member protection Member identification Archive maintenance Change history Disaster recovery Authority of each CCB over listed configuration items	IEEE 828
7	Does status accounting include Data collection Identified reports Problem investigation authority Maintaining and reporting Status of specifications Status of changes Status of product versions Status of software updates Status of client-furnished items	IEEE 828

Table A-11
Software Configuration Management Criteria (cont.)

8	Are suppliers of software products (e.g., COTS) under control? For each supplier. . . Is the SCM capability known? How is SCM performance monitored? For each product. . . Is the version in use archived? Is the version ID'd & baselined? Is the product under change control? Are product interfaces under control? Are suppliers CM audits "visible?" Is there valid problem tracking? Regarding supplier records. . . What records are kept? Can you get at them? How good are they? What security does the supplier have?	IEEE 828 and IEEE 1042. "The supplier (is responsible for) validation, storage, protection, and maintenance of (included software products)." ISO 9000-3
9	Are the records to be maintained identified and are there retention periods specified for each type of record?	IEEE 828
10	What additional policies and directives govern the configuration management?	See Table A-21 for a list of typical policies and directives

Table A-12
Software V&V Criteria

1	Does the V&V plan cover the minimum required subjects in the required format?	Format and subject matter is standard-dependent, but most standards specify similar approaches. See IEEE 1012
2	Is the organizational structure of the V&V function described, including the independence (or lack thereof) of the V&V organization from the software development organization?	IEEE 1012
3	Have the minimum required V&V tasks been accomplished?	See Table A-19 for minimum tasks
4	Does the V&V function detect errors as early in the development process as possible?	IEEE 1012
5	Can software changes and their consequences be assessed quickly?	IEEE 1012
6	Are V&V functions coordinated with the software development life cycle?	IEEE 1012
7	Are significant portions of V&V data missing?	See Table A-13

Table A-13
Actions to Take When Data is Missing

1	Can missing data be reconstructed from other available data?	Reconstruct data (see Table A-20) and proceed to Table A-12. ANSI/ANS-10.4
2	Can missing data be reverse-engineered from existing software products?	Reverse-engineer data (see Table A-20) and proceed to Table A-12. ANSI/ANS-10.4
3	Is recovered data and/or usage experience and configuration control insufficient to justify intended usage?	See Table A-20 for minimum data. If insufficient, terminate with prejudice. ANSI/ANS-10.4 and IEEE 828
4	Is sufficient test data available to support intended usage?	Reconstruct tests and proceed to Table A-12. ANSI/ANS-10.4

Table A-14
Minimum SQA Documentation

1	Software Quality Assurance Plan	IEEE 730.1
2	Software Requirements Specification	IEEE 730.1
3	Software Design Description	IEEE 730.1
4	Software V&V Plan	IEEE 730.1
5	Software V&V Report	IEEE 730.1
6	User Documentation (Manuals)	IEEE 730.1
7	Software Configuration Management Plan	IEEE 730.1

Table A-15
Minimum Required SQA Reviews and Audits

1	Software Requirements Review	IEEE 730.1
2	Preliminary Design Review	IEEE 730.1
3	Conceptual Design Review	IEEE 730.1
4	Software V&V Plan Review	IEEE 730.1
5	Functional Audits (e.g., validations)	IEEE 730.1
6	Physical Audits (e.g., physical deliverables)	IEEE 730.1
7	In-process Audits (e.g., life cycle stage verification audits)	IEEE 730.1
8	Managerial Reviews	IEEE 730.1

Table A-16
SQA, SCM, and V&V for Other Software Suppliers

1	SQA for purchased product shall meet the same requirements as if it were developed in-house. For to-be-developed COTS, the other software supplier shall perform the requirements of IEEE 730.1. For previously developed COTS, the "methods used to assure the suitability of the product for (its intended) use" shall be described.	IEEE 730.1
	"Software suppliers" shall select subcontractors on the basis of their ability to meet subcontract requirements, including quality requirements.	ISO 9000-3
2	SCM for purchased product shall meet the same requirements as if it were developed in-house. As a minimum, the other software supplier is required to implement the provisions of IEEE 828.	IEEE 828. See also Table A-11, line 8
3	V&V for COTS is not addressed, except indirectly through IEEE 730.1 through its provision requiring IEEE 730.1 compliance of the software supplier, or through ANSI/ANS-10.4 through its provisions for reconstruction of missing data.	See Table A-10, line 8, and Table A-13

Table A-17
Suggested Additional Documentation

1	Software Development Plan	IEEE 730.1
2	Standards & Procedures Manual	IEEE 730.1
3	Software Project Management Plan	IEEE 730.1
4	Software Maintenance Manual	IEEE 730.1
5	User Requirements Specification	IEEE 730.1
6	External Interfaces Specification	IEEE 730.1
7	Internal Interfaces Specification	IEEE 730.1
8	Operations Manual	IEEE 730.1
9	Installation Manual	IEEE 730.1
10	Training Manual	IEEE 730.1
11	Training Plan (for SQA personnel)	IEEE 730.1
12	Software Metrics Plan	IEEE 730.1
13	Software Security Plan	IEEE 730.1

Table A-18
Suggested Areas of Standardization

1	Documentation Standards	IEEE P730.2
2	Logical Structure Standards	IEEE P730.2
3	Coding Standards	IEEE P730.2
4	Comment Standards	IEEE P730.2
5	Testing Standards	IEEE P730.2
6	SQA Product & Process Metrics	IEEE P730.2

Table A-19
Minimum V&V Tasks

1	Software V&V Plan	IEEE 730.1 and IEEE 1012
2	Requirements (e.g., software requirements specification (SRS)) Analysis Existence Clarity Consistency Completeness All functions included Environment specified Inputs & outputs specified Standards used specified Correctness Feasibility Testability	IEEE 1012 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4
3	SRS Traceability Analysis	IEEE 1012 & ANSI/ANS-10.4
4	Interface Requirements Analysis	IEEE 1012 & ANSI/ANS-10.4
5	Test Plan Generation	IEEE 1012 & ANSI/ANS-10.4
6	Acceptance Test Plan Generation	IEEE 1012
7	Design Analysis Completeness Correctness Consistency Clearness Feasibility	IEEE 1012 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4
8	Design Traceability Analysis	IEEE 1012 & ANSI/ANS-10.4
9	Interface Design Analysis	IEEE 1012
10	Unit Test Plan Generation	IEEE 1012 & ANSI/ANS-10.4
11	Integration Test Plan Generation	IEEE 1012 & ANSI/ANS-10.4
12	Test Designs Code test drivers	IEEE 1012 ANSI/ANS-10.4
13	Source Code Analysis Conformance to standards Adequate comments Clear and understandable Consistent with design Strong typing Error-checking	IEEE 1012 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4 ANSI/ANS-10.4
14	Source Code Traceability	IEEE 1012
15	Interface Code Analysis Well-controlled software interfaces	IEEE 1012 ANSI/ANS-10.4
16	Documentation Evaluation	IEEE 1012
17	Test Procedure Generation Unit Test Integration Test System Test Acceptance Test	IEEE 1012 & ANSI/ANS-10.4

Table A-19
Minimum V&V Tasks (cont.)

18	Unit Test Execution	IEEE 1012
	Unit test results	ANSI/ANS-10.4
19	Integration Test Execution	IEEE 1012
	Size	ANSI/ANS-10.4
	Timing	ANSI/ANS-10.4
	Interface control	ANSI/ANS-10.4
	Interactions verified	ANSI/ANS-10.4
	Build control and documentation	ANSI/ANS-10.4
20	System Test Execution	IEEE 1012
	Each requirement tested?	ANSI/ANS-10.4
	Each requirement met?	ANSI/ANS-10.4
	All test cases executed and checked?	ANSI/ANS-10.4
21	Acceptance Test Execution	IEEE 1012
22	Installation Configuration Audit	IEEE 1012
	Deliverables identified	ANSI/ANS-10.4
	Can delivered program be rebuilt?	ANSI/ANS-10.4
	Do test cases still work?	ANSI/ANS-10.4
23	V&V Final Report	IEEE 1012
24	Baseline Change Assessment (as required)	IEEE 1012
25	Review Support—participation in software and management reviews	IEEE 1012

Table A-20
Minimum Documentation Needed for *a Posteriori* V&V

1	Problem statement	ANSI/ANS-10.4
2	Requirements specification	ANSI/ANS-10.4
3	Design specification	ANSI/ANS-10.4
4	Test plan and test results	ANSI/ANS-10.4

Table A-21
Typical Policies and Directives of a Configuration Management Operation

1	Definition of software levels or classes	IEEE 828
2	Naming conventions	IEEE 828
3	Version ID conventions	IEEE 828
4	Product ID policy	IEEE 828
5	IDs of specs, test plans, manuals & documents	IEEE 828
6	Media ID and file management	IEEE 828
7	Documentation release process	IEEE 828
8	Software release to general library	IEEE 828
9	Problem reports, change requests and orders	IEEE 828
10	Structure & operation of CCBs	IEEE 828
11	Acceptance or release of software products	IEEE 828
12	Operating rules for the software library	IEEE 828
13	Audit policy	IEEE 828
14	Methods for CCB assessment of change impact	IEEE 828
15	Level of testing or assurance required before an item is accepted for configuration management—may be related to software classes	IEEE 828
16	Level of SQA or V&V required before an item is accepted for configuration management—may be related to software classes	IEEE 828